Model-Based Useware Engineering

Dr. Gerrit Meixner
Senior Researcher

W3C Workshop on Future Standards for Model-Based User Interfaces
Rome, Italy, 13.-14. May 2010
April 2010:
- 753 employees (overall)
- 260 Researcher
- 116 ongoing projects

„DFKI is the world‘s largest and leading Center of Excellence for basic and application-oriented AI research.“
Living Lab - SmartFactory

- First multi vendor research, development and demonstration center for industrial ICT
- **Goal:** The integration of mature ICT into factory automation
MBUID Use Cases

1. Industrial Projects: Usability Engineering
   - @Development-time
   - User-centered development process +
     Model-based user interface development methodology
     ➢ Supporting project staff with tools (e.g. prototyping, code generators)

2. Configuration and maintaining of industrial devices
   - @Run-time
   - Automatic UI generation
   - Adaptive UIs
     ➢ Ad-hoc access to devices via one universal interaction device (instead of using many different devices)
Useware Engineering Process

- Incorporates users and clients into all project phases (Iterative)
- Different (overlapping) main phases:
  - **Analysis**: Understanding the users, their tasks and the context-of-use
  - **Structuring**: Deduction of a single, harmonized task structure
  - **Design**: Deduction of abstract & concrete UIs
  - **Evaluation**: Iterative testing of mock-ups/prototypes with users

→ Approved in many different research- and industrial projects since 1991
Different layers of a user interface

- Analysis
  - Task model
- Structuring
  - Use model
- Design
  - Abstract UI model
  - Concrete UI model
- Realisation
  - Final UI (source code)
Useware Markup Language (useML) 2.0

- Different tasks types (e.g. system task, interactive task)
- Elementary use objects --> more detailed specification of interactive tasks
- Optionality and Cardinality
- Logical and temporal conditions (pre-conditions, invariants, post-conditions)
- 5 Temporal Operators
Udit – useML-Editor (1/3)

Features:
- Udit supports the whole expressiveness of useML 2.0
- Integrated semantic model checker (e.g. warning, error)
- Multilingual (German/ English/ …)
- Export of use models (DISL, figure, direct print)
- Expand nodes (display / hide sub-tasks)
- 3 level of detail
Udit – useML-Editor (2/3)

- Project-specific adaptation is possible (user group, access location, device type, etc.)
Udit – useML-Editor (3/3)
Example of our MBUID-approach

Task & concepts (useML)

Abstract UI (DISL)

Concrete UI (UIML)

Final UI (Java/Swing)
MBUID@Run-time
Future Ambient Intelligent Production Environments still consist of many Actors, Sensors …

→ Devices have to be maintained
Devices have to be maintained

Today’s problems…

- Bad Accessibility
  E.g. many devices are located > 2m above the floor
MBUID@Run-time

- Devices have to be maintained

Today’s problems…

- **Bad Accessibility**
  E.g. many devices are located > 2m above the floor

- **Minimalistic User Interfaces**
  UIs usually consist of small displays and few buttons
MBUID@Run-time

Devices have to be maintained

Today’s problems…

- **Bad Accessibility**
  E.g. many devices are located > 2m above the floor

- **Minimalistic User Interfaces**
  UIs usually consist of small displays and few buttons

- **Proprietary User Interfaces**
  Each UI has its’ own menu structure, layout, behavior…
SmartMote: A remote control for Intelligent Production Environments

Key Features:

✔ Task-centered
   Each UI is described in a use model, describing the users tasks

✔ Adaptive
   The UI is generated at run-time and adapts to the users context-of-use (e.g. access location)

✔ Wireless
   The SmartMote uses Bluetooth & WLAN for a seamless device communication
Indoor Positioning Systems installed in the SmartFactoryKL

- Ubisense UWB-Realtime Positioning System
  ![Ubisense](image)
- RFID Grid for Mobile Workshop Navigation
  ![RFID Grid](image)
- Cricket Ultrasonic Indoor Location System
  ![Cricket](image)
Improving the quality of the generated UI

Using HCI-patterns to improve usability of run-time generated UIs

- HCI-Patterns are **proven solutions for recurring problems**
- Focus on **When**, **Why** and **How** a solution should be applied
- Primary domain: **webdesign**

**So, what’s the deal?!**

- HCI-Patterns still lack in formalization!

- Run-Time generation demands for a machine-processable pattern form
• Overall aim: maintain principles of good GUI design
• Problem: guidelines are often not followed or interpreted correctly
• Reasons:
  – too abstract → no instant answers to the designer’s questions
  – too complex → too difficult to understand
  – too expansive → too time consuming to read
→ Solution: automatic verification of GUI guidelines
Thanks for your attention
Contact

Dr. Gerrit Meixner

E-Mail: Gerrit.Meixner@dfki.de
Web: http://www.dfki.de
Phone: +49 (0)631 205-3707
Fax: +49 (0)631 205-3705

German Research Center for Artificial Intelligence (DFKI)
Innovative Factory Systems (IFS)
PO Box 3049
D-67653 Kaiserslautern